Knowl Inf Syst
DOI 10.1007/s10115-013-0679-x

REGULAR PAPER

# Explaining prediction models and individual predictions with feature contributions

**Erik Štrumbelj · Igor Kononenko**

Received: 12 November 2012 / Revised: 2 August 2013 / Accepted: 17 August 2013


**Abstract** We present a sensitivity analysis-based method for explaining prediction models that can be applied to any type of classification or regression model. Its advantage over existing general methods is that all subsets of input features are perturbed, so interactions and redundancies between features are taken into account. Furthermore, when explaining an additive model, the method is equivalent to commonly used additive model-specific methods. We illustrate the method's usefulness with examples from artificial and real-world data sets and an empirical analysis of running times. Results from a controlled experiment with 122 participants suggest that the method's explanations improved the participants' understanding of the model.

**Keywords** Knowledge discovery · Data mining · Visualization · Interpretability · Decision support

## 1 Introduction

Prediction models are an important component of decision support systems. Applications range from credit scoring [11] and fraud detection [5] to financial auditing [4] and efficiency analysis [18]. In such applications, model interpretability is often as important if not more important than prediction accuracy.

Some more difficult to interpret models require additional post-processing to (a) obtain a better understanding of the model and (b) increase the end-user's level of trust in the model. The latter is especially important in risk-sensitive domains such as finance and medicine, where experts are reluctant to trust prediction model's predictions without an additional explanation.

E. Štrumbelj (✉) · I. Kononenko
Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25,
1000 Ljubljana, Slovenia
e-mail: erik.strumbelj@fri.uni-lj.si

Published online: 30 August 2013

Most explanation methods are model-specific. Some models, such as decision and regression trees, rules [6,8], and nearest neighbors-based methods, are self-explanatory. Complex models, such as artificial neural networks and SVM (support vector machines), received the most attention, because they are often very successful but difficult to interpret (see [28], and references therein).

Linear regression and other additive models can be additionally explained by plotting the marginal effect of each input variable. For additive models, a prediction is the sum of individual marginal effects, which makes such visualizations a tool for graphical computation of predictions—a nomogram. This fact has been exploited to provide an explanation for the naive Bayes classifier [3,17,23], linear SVM [14], logistic regression [21], Cox regression models [15], and additive models in general [26].

This paper focuses on explanation methods that can be applied to prediction models of any type. Such general approaches must treat the model as a black box and are thus restricted to changing the inputs and observing the changes in the output. While not being able to exploit the specifics of the model, such methods have the advantage of being applicable to any type of model. This facilitates comparison of different types of models and, in practical applications, eliminates the need to replace the explanation method when the underlying model is replaced.

The key component of general explanations is the contributions of individual input features. A prediction is explained by assigning to each feature a number which denote its influence. For each feature, such contributions can be aggregated to plot the feature's average contribution against the feature's value. This provides an overview of the model and is similar to plotting the marginal effect for an additive model.

We begin with a simple illustrative example—a linear regression model:

$$f(x) = f(x_1, \ldots, x_n) \approx y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n.$$

If the input features are standardized, the coefficient $\beta_i$ can be interpreted as the $i$th feature's global importance. However, in practice, we are more interested in how a particular value influences the prediction. We turn to the following expression

$$\varphi_i(x) = \beta_i x_i - \beta_i E[X_i], \tag{1}$$

which is also known as the situational importance of $X_i = x_i$ [1].

The situational importance is the difference between what a feature contributes when its value is $x_i$ and what it is expected to contribute. If the situational importance is positive, then the feature has a positive contribution (increases the prediction for this particular instance), if it is negative, then the feature has a negative contribution (decreases the prediction), and if it is 0, it has no contribution.

To illustrate, observe the linear model $f(x_1, x_2) = 2x_1 - 3x_2 + 4$, with both input features uniformly distributed on $[-1, 1]$. How much do the two input features contribute for the prediction $f(\frac{1}{2}, \frac{1}{3})$? For this instance, the situational importance of the first and second feature are 1 and $-1$, respectively. Therefore, the first feature contributes positively and the second one negatively.

We can plot the average situational importance $\psi$ of every value, to obtain an overview of how each feature contributes across all of its values (see Fig. 1). This plot not only shows how different feature values contribute, but it can also be used to semi-graphically compute the prediction for any instance. As previously mentioned, such a plot can be produced for any additive model, a fact that has been exploited in developing several model-specific explanation methods [14,15,21,23,26].

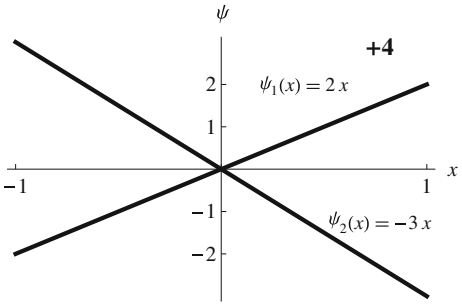


**Fig. 1** The marginal effects of the two input features for the model $f(x_1, x_2) = 2x_1 - 3x_2 + 4$. Both input features' functions are shown on the same graph. Such plots can be used for semi-graphical computation of the model's prediction for an arbitrary instance $x = < x_1, x_2 >$. For each input feature, we use the plotted function to map its value from the $x$-axis to that value's contribution on the $y$-axis. All that remains is to sum the two contributions and the expectation $E[f] = 4$

Computing the contributions for our illustrative example was simple, because the model is known and the features do not interact. Therefore, the contribution of some $x_i$ is the same across all instances, regardless of the values of other features. In our problem setting, however, the model is unknown and no assumptions are made other than that the model maps from some known input feature space to a known codomain. These restrictions are necessary for the method to be general, but restrict us to changing the inputs and observing the outputs.

Previous general approaches [19,24] tackled the problem in the following way:

$$\varphi_i(x) = f(x_1, \ldots, x_n) - E[f(x_1, \ldots, X_i, \ldots, x_n)] \tag{2}$$

Equation (2) is the difference between a prediction for an instance and the expected prediction for the same instance if the $i$th feature had not been known. In practice, expression Eq. (2) is not difficult to approximate (or compute, if the feature's domain is finite)—we have to perturb the values of the $i$th feature, while the values of other input features remain fixed, and then average the prediction. Additionally, if $f$ is an additive model, Eq. (2) is equivalent to Eq. (1), so in the case of an additive model, we do not lose any of the previously mentioned advantages associated with explaining an additive model.

However, when the model is not additive, as it is often the case in practical applications, the approach gives undesirable results. For example, observe the model $f(x_1, x_2) = x_1 \vee x_2$, where both features are uniformly distributed on $\{0, 1\}$. When computing the contribution of the first feature for $f(1, 1) = 1$, we see that perturbing its value does not change the prediction—the first features contribution is 0. The same holds for the second feature. Therefore, both features get a 0 contribution, which is undesirable.

We learn from the previous example that perturbing one feature at a time gives undesirable results and that all subsets of features have to be perturbed to avoid such issues. In our previous work, we proposed a general method for computing the situational importance for classification and, separately, regression models that dealt with the aforementioned shortcomings of existing general explanation methods [27–29].

This paper builds on the method from our previous work. The general method for computing the situational importance of features is modified and extended to marginal importance of feature values and feature importance. Appropriate sampling algorithms are provided, and we discuss two extensions that improve the algorithms efficiency. We also show that in the case of additive models, the proposed method is equivalent to the explanation commonly

used for additive models. In the experimental part of the paper, we provide an empirical analysis of running times, illustrative examples, and the results of a controlled experiment of the usefulness of the contribution-based instance explanations.

The remainder of the paper is organized as follows. In Sect. 2, we provide the essential background from [28] and [29]. In Sect. 3, we describe and improve the approximation algorithm for computing a feature's contribution for an instance [28] and the average contribution of a feature's value. Section 4 is dedicated to experimental results and visual inspection of instance and model visualizations. Section 5 concludes the paper.

## 2 Computing a feature's contribution

The following notation will be used throughout the paper. Let $\mathcal{X} = [0, 1]^n$ be our feature space, $Y$ the target variable, and $\{y_i; x_{1,i}, x_{2,i}, \ldots, x_{n,i}\}_{i=1}^{M}$ a data set of $M$ instances. The function $f : \mathcal{X} \rightarrow \Re$ represents the model that is used to predict the value of the target variable for an instance $x \in \mathcal{X}$. In the classification case, we take the one-vs-all approach—we choose and observe the prediction for a single class value. Any class value of interest can be chosen or explanations can be generated for several values. We will refer to the chosen value as the point-of-view class.

First, observe how a feature's value contributes for a simple linear model. That is, let us assume for a moment that $f$ takes the form

$$f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n.$$

The contribution of the $i$th feature's value for some instance $x \in \mathcal{X}$ is the difference between the model's prediction and the expected prediction if the $i$th feature's value is not known:

$$\begin{aligned} \varphi_i^{additive}(x) &= \beta_0 + \cdots + \beta_i x_i + \cdots + \beta_n x_n - (\beta_0 + \cdots + \beta_i \mathbb{E}[x_{,i}] + \cdots + \beta_n x_{,n}) \\ &= \beta_i (x_i - E[X_i]). \end{aligned} \tag{3}$$

The contribution in Eq. (3) is sometimes also referred to as the situational importance of $x_i$ [1]. Observe how such a contribution is independent of the values of other features. This is due to the fact that the linear model is additive (that is, the features do not interact). This property makes the linear model and other additive models easy to interpret.

In practice, features often interact. To avoid the shortcomings of existing methods, we have to take into account every subset of features. We generalize Eq. (3) by first defining the model's prediction conditional to only a subset of features' values being known:

$$f_Q(x) = \mathbb{E}[f | X_i = x_i, \forall i \in Q], \tag{4}$$

where $Q \subseteq S = \{1, 2, \ldots, n\}$ is a subset of features. For an empty set, Eq. (4) reduces to $f_{\{\}}(x) = \mathbb{E}[f]$. Eq. (4) allows us to define the contribution of a subset of feature values:

$$\Delta_Q(x) = f_Q(x) - f_{\{\}}(x). \tag{5}$$

Equation (5) is the change in prediction caused by observing the values of a certain subset of features for some instance $x \in \mathcal{X}$. To provide a contribution similar to the one for the linear model, we have to map these $2^n$ terms into $n$ contributions, one for each feature's value. First, we implicitly define interactions by saying that the contribution of a subset of feature values is the sum of all interactions across all subsets of those feature values:

$$\Delta_Q(x) = \sum_{W \subseteq Q} \mathcal{I}_W(x), \tag{6}$$

which uniquely determines the interactions:

$$\mathcal{I}_Q(x) = \Delta_Q(x) - \sum_{W \subset Q} \mathcal{I}_W(x). \tag{7}$$

Finally, each interaction is divided among the participating feature values, which defines the contribution:

$$\varphi_i(x) = \sum_{W \subseteq S \setminus \{i\}} \frac{\mathcal{I}_{W \cup \{i\}}(x)}{|W| + 1}. \tag{8}$$

This leads to the following explicit definition (see [28] for proof):

$$\varphi_i(x) = \sum_{Q \subseteq S \setminus \{i\}} \frac{|Q|!(|S| - |Q| - 1)!}{|S|!} (\Delta_{Q \cup \{i\}}(x) - \Delta_Q(x)). \tag{9}$$

Equation (9) is equivalent to the Shapley value [25], a concept from coalitional game theory. In a coalitional game, it is usually assumed that $n$ players form a grand coalition that has a certain worth (in our case, $\Delta_S$). We also know how much each smaller (subset) coalition would have been worth ($\Delta_Q$, $Q \subset S$). The goal is to distribute the worth of the grand coalition among players in a fair way (that is, each player should receive his fair share, taking into account all sub-coalitions). The Shapley value is one such solution, and it is the unique solution that satisfies the following properties [25]:

- $\sum_{i=1}^{n} \varphi_i(x) = \Delta_S(x)$
- $\forall W \subseteq S \setminus \{i\} : \Delta_W = \Delta_{W \cup \{i\}} \Rightarrow \varphi_i(x) = 0$
- $\forall W \subseteq S \setminus \{i, j\} : \Delta_{W \cup \{i\}} = \Delta_{W \cup \{j\}} \Rightarrow \varphi_i(x) = \varphi_j(x)$
- $\forall x, y \in \mathcal{X}$: $\varphi(x + y) = \varphi(x) + \varphi(y)$, where $\Delta_Q(x + y) = \Delta_Q(x) + \Delta_Q(y)$ for all $Q \subseteq S$

In the context of our explanation with local contributions, the properties have the following interpretation. The contributions are implicitly normalized, which makes them easier to interpret and compare. If a feature's value does not have any impact on the prediction, then it will be assigned a 0 contribution. If two features' values have the a symmetrical impact across all subsets, they will be assigned equal contributions, and the local contributions are additive across instances.

## 3 Approximation algorithm

Computing Eq. (9) has an exponential time complexity, which makes the method infeasible for practical use. The following approximation is used to reduce the computational complexity. We start by writing a different but equivalent formulation of Eq. (9):

$$\varphi_i(x) = \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \left( \Delta_{Pre^i(\mathcal{O}) \cup \{i\}} - \Delta_{Pre^i(\mathcal{O})} \right), \qquad i = 1, \ldots, n, \tag{10}$$

where $\pi(n)$ is the set of all ordered permutations of the feature indices $\{1, 2, \ldots, n\}$ and $Pre^i(\mathcal{O})$ is the set of all indices that precede $i$ in permutation $\mathcal{O} \in \pi(n)$.

If the cost of computing the $\Delta$-terms would be zero, Eq. (10) could be approximated using a simple sampling algorithm, where $\left( \Delta_{Pre^i(\mathcal{O}) \cup \{i\}} - \Delta_{Pre^i(\mathcal{O})} \right)$ would be one sample (see, for example, [7]). However, the computational complexity of computing the $\Delta$-terms is exponential. As shown in [29], it is sufficient to limit ourselves to such distributions

of instances $p$ that individual features are distributed independently. Now, Eq. (5) can be simplified into the following:

$$\begin{aligned}\Delta_Q(x) &= f_Q(x) - f_{\{\}}(x)\\ &= \sum_{w\in\mathcal{X};\forall i:(w_i=x_i \vee i\notin S)} p(w)f(w) - \sum_{w\in\mathcal{X};\forall i:(w_i=x_i)} p(w)f(w)\\ &= \sum_{w\in\mathcal{X}} p(w)(f(w_{[w_i=x_i,i\in S]}) - f(w)),\end{aligned} \tag{11}$$

where the notation $w_{[w_i=x_i,i\in S]}$ denotes instance $w$ with the value of feature $i$ replaced with that feature's value in instance $x$, for each $i \in S$. For example, with $w = \langle 2, 4, 6\rangle$ and $x = \langle 3, 5, 7\rangle$, $w_{[w_i=x_i,i\in\{1,3\}]} = \langle 3, 4, 7\rangle$.

The $\Delta$-terms in Eq. (10) are replaced with Eq. (11) to obtain:

$$\varphi_i(x) = \frac{1}{n!}\sum_{\mathcal{O}\in\pi(N)}\sum_{w\in\mathcal{X}} p(w)\cdot\left(f(w_{[w_j=x_j,j\in Pre^i(\mathcal{O})\cup\{i\}]}) - f(w_{[w_j=x_j,j\in Pre^i(\mathcal{O})]})\right), \tag{12}$$

The following sampling procedure is used. Let

$$V_{\mathcal{O},w\in\mathcal{X}} = \left(f(w_{[w_j=x_j,j\in Pre^i(\mathcal{O})\cup\{i\}]}) - f(w_{[w_j=x_j,j\in Pre^i(\mathcal{O})]})\right),$$

for all permutation/instance pairs, be the sampling population. When sampling at random and with replacement, we draw sample $V_{\mathcal{O},w\in\mathcal{X}}$ with probability $p(w)$. Draw $m$ such samples $V_1, V_2, \ldots, V_m$ at random with replacement and define

$$\hat{\varphi}_i = \frac{1}{m}\sum_{j=1}^{m} V_j, \tag{13}$$

It follows that $\hat{\varphi}_i$ is approximately normally distributed with mean $\varphi_i$ and variance $\frac{\sigma_i^2}{m}$, where $\sigma_i^2$ is the population variance. That is, $\hat{\varphi}_i$ is an unbiased and consistent estimator of $\varphi_i(x)$. The described approximation algorithm is summarized in Algorithm 1.

### 3.1 Quasi-random and adaptive sampling

We considered two improvements that increase the efficiency of the approximation algorithm. First, the approximation algorithm is a form of Monte Carlo integration. Therefore, for faster convergence, quasi-random sampling can be used instead of pseudo-random sampling. In our experiments, we used the Sobol low-discrepancy quasi-random sequence [13].

Second, to compute the explanation for an instance $x$, we need to compute the contribution for each of the $n$ features for that instance. In practice, we want to do this in a controlled amount of time to minimize the overall approximation error. The approximation error of the estimator $\varphi_i(x)$ depends on the population variance which may not be the same for all features. Given that in practice, we are limited to a certain number of samples $m$, it makes sense to adapt $m_i$ the number of samples drawn for a feature to that feature's variance $\sigma_i^2$. We discuss two cases—minimizing the squared $\sum_{i=1}^{n}(\hat{\varphi}_i - \varphi_i)^2$ and the absolute approximation error $\sum_{i=1}^{n}|\hat{\varphi}_i - \varphi_i|$.

Recall that the estimate $\hat{\varphi}_i$ is approximately normally distributed $\hat{\varphi}_i \approx N(\varphi_i, \frac{\sigma_i^2}{m_i})$. It follows that $\hat{\varphi}_i - \varphi_i \approx N(0, \frac{\sigma_i^2}{m_i})$. The distribution of the absolute error for the $i$-th feature

**Algorithm 1** Approximating the $i$th features contribution for model $f$, instance $x \in \mathcal{X}$ and distribution $p$. Draw $m$ samples

$\varphi_i(x) \leftarrow 0$
**for** 1 to $m$ **do**
  select, at random, permutation $\mathcal{O} \in \pi(n)$
  select, at random, $w \in \mathcal{X}$
  construct two instances:

$$\vec{b}_1 \leftarrow \overbrace{\boxed{\text{preceding } i\text{-th in } \mathcal{O}}}^{\text{take values from } x}\ \boxed{i}\ \overbrace{\boxed{\text{succeeding } i\text{-th in } \mathcal{O}}}^{\text{take values from } w}$$

$$\vec{b}_2 \leftarrow \overbrace{\boxed{\text{preceding } i\text{-th in } \mathcal{O}}}^{\text{take values from } x}\ \boxed{i}\ \overbrace{\boxed{\text{succeeding } i\text{-tega v } \mathcal{O}}}^{\text{take values from } w}$$

  $\varphi_i(x) \leftarrow \varphi_i(x) + f(\vec{b}_1) - f(\vec{b}_2)$

**end for**
$\varphi_i(x) \leftarrow \frac{\varphi_i(x)}{m}$

$Z_i = |\hat{\varphi}_i - \varphi_i|$ is half-normal, with $E[Z_i] = \sqrt{\frac{\sigma_i^2}{m_i}}\sqrt{\frac{2}{\pi}}$. The expectation for the sum of absolute errors is

$$E\left[\sum_{i=1}^{n} Z_i\right] = \sum_{i=1} n\sqrt{\frac{\sigma_i^2}{m_i}}\sqrt{\frac{2}{\pi}} = \sqrt{\frac{2}{\pi}}\sum_{i=1}^{n}\frac{\sigma_i}{\sqrt{m_i}}.$$

Similarly, for the sum of squared errors, we take $Z_i \approx (\hat{\varphi}_i - \varphi_i)^2$. The expectation $E[Z_i^2] = Cov[Z_i, Z_i] + 2E[Z_i] = \frac{\sigma_i^2}{m_i}$. The expectation for the sum of absolute errors is

$$E\left[\sum_{i=1}^{n} Z_i\right] = \sum_{i=1}^{n}\frac{\sigma_i^2}{m_i}.$$

In practice, we first take samples for each input feature to obtain an initial estimate of $\sigma_i$. After the minimum amounts of samples have been taken, the goal is to distribute $m_{max}$, the total number of samples we can compute, among individual features a way that we minimize the expected error. Regardless of which error we use, a greedy approach is optimal. That is, if the current amount of samples taken for each feature are $m_1, \ldots, m_n$, then we should take the sample for the feature that maximizes $\sqrt{\frac{\sigma_i^2}{m_i}} - \sqrt{\frac{\sigma_i^2}{m_i+1}}$ (or $\frac{\sigma_i^2}{m_i} - \frac{\sigma_i^2}{m_i+1}$). This is a direct consequence of the fact that functions $g(z) = \frac{\sigma_i}{\sqrt{z}}$ and $g(z) = \frac{\sigma_i^2}{z}$ are both strictly decreasing on $z \in \Re^+$. Therefore, the currently best choice is also better than all possible future choices, regardless of the order in which future samples are taken.

The adaptive sampling version of the algorithm is summarized in Algorithm 2. Note that, we used Knuth's incremental algorithm for computing the variance [16,30].

### 3.2 Average contribution of a feature's value

The plots mentioned in the Introduction are a common and efficient way of presenting an overview of additive models (see Fig. 1). Such plots show, for each feature, the function that

**Algorithm 2** Approximating all features' contributions for model $f$, instance $x \in \mathcal{X}$ and distribution $p$. Draw $m_{min}$ samples for each feature, draw a total of $m_{max} >= n \cdot m_{min}$ samples

**for** $i = 1$ to $n$ **do**
  $m_i \leftarrow 0, \varphi_i(x) \leftarrow 0$
**end for**
**while** $\sum_{i=1}^{n} m_i < m_{max}$ **do**
  **if** $\forall i : m_i \geq m_{min}$ **then**
    pick a $j \in \{1..n\}$ which maximizes $(\sqrt{\frac{\sigma_j^2}{m_j}} - \sqrt{\frac{\sigma_j^2}{m_j+1}})^{\dagger}$
  **else**
    pick a $j$, such that $m_j < m_{min}$
  **end if**
  $\varphi_i(x) \leftarrow \varphi_i(x) +$ (result of Algorithm 1 for $j$-th feature and $m = 1$)
  update $\sigma_i^2$ using an incremental algorithm.
  $m_j \leftarrow m_j + 1$
**end while**
**for** $i = 1$ to $n$ **do**
  $\varphi_i(\vec{a}) \leftarrow \frac{\varphi_i(a)}{m_i}$
**end for**

† if minimizing the squared error, maximize $(\frac{\sigma_j^2}{m_j} - \frac{\sigma_j^2}{m_j+1})$ instead

maps its values to situational contributions of those values. Recall that, for additive models, where the features do not interact, the situational contribution $\psi_i(q)$ of the $i$th feature's value $q \in \mathcal{X}_i$ is the same for all instances. However, if the model is not additive and the features interact, then the situational contribution of a feature's value depends on the values of other features.

To produce a similar plot, we average the $i$th feature's value $q$'s local contributions across all instances with that value:

$$
\begin{aligned}
\psi_i(q) &= \sum_{x \in \mathcal{X} \wedge x_i = q} p(x)\varphi_i(x) = \sum_{x \in \mathcal{X}} p(x)\varphi_i(x_{[x_i=q]}) \\
&= \sum_{x \in \mathcal{X}} p(x) \left( \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \sum_{w \in \mathcal{X}} p(w) \left( f(w_{[w_j = x_j, j \in Pre^i(\mathcal{O}); w_i = q]}) \right.\right. \\
&\quad \left.\left. - f(w_{[w_j = x_j, j \in Pre^i(\mathcal{O})]}) \right) \right) \\
&= \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \left( \sum_{x \in \mathcal{X}} \sum_{w \in \mathcal{X}} p(x)p(w) \left( f(w_{[w_j = x_j, j \in Pre^i(\mathcal{O}); w_i = q]}) \right.\right. \\
&\quad \left.\left. - f(w_{[w_j = x_j, j \in Pre^i(\mathcal{O})]}) \right) \right) \\
&= \frac{1}{n!} \sum_{\mathcal{O} \in \pi(N)} \sum_{x \in \mathcal{X}} p(x) \left( f(x_{[x_i=q]}) - f(x) \right) \\
&= \sum_{x \in \mathcal{X}} p(x) \left( f(x_{[x_i=q]}) - f(x) \right). \qquad (14)
\end{aligned}
$$

For the transition from line 3 to line 4 in Eq. (14), observe the probability of the composite instance $w_{[w_j=x_j, j\in Pre^i(\mathcal{O})]}$ being a particular instance $z$ if $w$ and $x$ are samples from $\mathcal{X}$. For a fixed permutation $\mathcal{O}$ and if independence of features is assumed, the probability of composing particular instance $z$ by composing two independent samples from $\mathcal{X}$ is $p(z)$, the probability of drawing at random instance $z$ from $\mathcal{X}$. Therefore, the term $\big(f(z_{[z_i=q]}) - f(z)\big)$ appears in the double summation with weight $p(z)$.

To compute Eq. (14), we use a similar approximation as before. We also compute the standard deviation of samples $\big(f(x_{[x_i=q]}) - f(x)\big)$, which can be interpreted as the input features overall importance for the model and is shown in the model visualizations in the Experimental Results in the form of a light gray line (see Fig. 5).

Let $\psi_i, i = 1 \ldots n$ be the average contribution functions. If $f$ is additive, then it holds for each input feature $i$ and its value $x$ that $\psi_i(q) = f_i(q) - E[f_i]$, where $f_i$ are the marginal effects of individual features:

$$\begin{aligned}
\psi_i(q) &= \sum_{x\in\mathcal{X}} p(x)(f(x_{[x_i=q]}) - f(x)) \\
&= \sum_{\vec{z}\in\mathcal{X}} p(x)(f_i(q) - f_i(x_i)) \\
&= \sum_{x\in\mathcal{X}} p(x) f_i(q) - \sum_{x\in\mathcal{X}} p(x) f_i(x_i) \\
&= f_i(q) - E[f_i].
\end{aligned}$$

That is, in the case of an additive model, the average contribution of a feature's value equals the situational contribution of that value.

## 4 Experimental evaluation

The experimental evaluation of the proposed method is divided into three parts. First, we preform a detailed analysis of running times across several well-known real-world data sets and artificial data sets using several different types of machine learning models. The purpose of this experiment is to see how the methods algorithm scales with an increasing number of features and to quantify the benefits of using the two proposed improvements (adaptive and quasi-random sampling).

Second, we apply the method to several different types of machine learning models trained on several different data sets. The purpose of this part is to provide illustrative examples, describe the visualizations, and point out where existing methods would fail to provide a reasonable explanation.

And third, we describe a controlled experiment with 122 student participants. Our goal was to measure the effect that explanations in the form of feature contributions have on a person's understanding of a prediction model.

The first two parts include experimentation on a number of different data sets and machine learning models. The artificial data sets used in these experiments are listed in Table 1 and are available as supplementary material. We also included several well-known regression and classification data sets: autoMpg, bodyfat, concrete, elevators, fishcatch, fruitfly, housing, machinecpu, pollution, stock, wine, and wisconsin(regression), anneal, breastCancerLJ, hepatitis, iris, monks1, monks2, monks3, mushroom, nursery, soybean, and zoo (classification). These data sets are available in *.arff* format from the Weka website (http://www.cs.

**Table 1** Number of instances (#I), total number of input features (#F), and brief description of artificial data sets

| Name | #I | #F | Description |
|---|---|---|---|
| *Classification* | | | |
| cChess | 2000 | 4 | Color of $4 \times 4$ chessboard point |
| cCondInd | 2000 | 8 | Conditionally independent features |
| cCross | 2000 | 6 | Even or odd quadrant in coordinate system |
| cDisjunctB | 2000 | 5 | Disjunction with binary input features |
| cDisjunctN | 2000 | 5 | Disjunction with numeric features |
| cGroup | 2000 | 4 | Clusters |
| cRandom | 2000 | 4 | Random input features |
| cRedundant | 2000 | 5 | Disjunction with redundant features |
| cSphere | 2000 | 5 | Point lies in the interior of a sphere |
| cXor | 2000 | 6 | Xor |
| *Regression* | | | |
| rDisjunctB | 2000 | 5 | Disjunction with binary input features |
| rDisjunctN | 2000 | 5 | Disjunction with numeric features |
| rLinear | 2000 | 5 | Linear problem |
| rLinNoisy | 2000 | 5 | Linear problem with noise |
| rLocLinear | 2000 | 5 | Locally linear problem |
| rNonLinPoly | 2000 | 5 | Third degree polynomial |
| rNonLinTrig | 2000 | 5 | Trigonometric function |
| rRandom | 2000 | 4 | Random input features |
| rRedundant | 2000 | 5 | Disjunction with redundant features |
| rXor | 2000 | 6 | Xor |

These data sets were constructed for the purpose of experimental verification of how general explanation methods perform on data with concepts such as disjunction, xor (exclusive or), conditionally independent features, redundant, and random features

waikato.ac.nz/ml/weka/). Most can also be found at the UCI Machine Learning Repository [9].

We included ten different variations of learning algorithms for classification and seven different variations for regression (see Table 2). All-used learning algorithms were from the Weka [10] machine learning software. Unless otherwise noted, default settings were used. All experiments were run on an off-the-shelf computer with a 2.4 GHz CPU and 2 GB of RAM.

### 4.1 Running times analysis

### 4.2 Sampling algorithm enhancements

We used the following procedure to measure the benefits of using adaptive sampling and quasi-random sampling. All regression data set/regression model and classification data set/classifier pairs were included in the experiment. For every such pair, we trained the model on 500 bootstrap samples and computed the mean-squared approximation error across all instances. We computed the error at different amounts of samples per feature and for all four combinations of the basic approximation algorithm and enhancements (both, just quasi-random, just adaptive, neither).

**Table 2** A list of learning algorithms that were included in the experiments

| Name | Description |
|---|---|
| AdaBoostM1 | Boosting with Naive Bayes or decision tree as base learner |
| Bagging | Bagging with either decision tree or regression tree as base learner |
| IBk | $k$-Nearest Neighbors with either $k = 1$ or $k = 11$) |
| J48 | Decision tree |
| LinearRegression | Linear regression |
| Logistic | Logistic regression |
| M5P | Regression tree |
| MultilayerPerceptron | Multi-layer artificial neural network with one hidden level |
| NaiveBayes | Naive Bayes classifier |
| SVO | Support Vector Machine with second degree polynomial kernel |
| SVMreg | Regression SVM |

IBk and MultilayerPerceptron were used for both regression and classification

The results shown in Fig. 2a suggest that both enhancements improve the efficiency of the algorithm. That is, fewer samples are needed to achieve the same approximation error. The improvement achieved with quasi-random sampling is small compared to the improvement achieved by adaptive sampling. Best results are achieved when both enhancement are used.

### 4.3 Scalability

We illustrate how the method scales with an increasing number of features on two additional data sets. The *linear50* data set is a regression problem with 1,000 instances and 50 standardized numerical input features. The class value is a linear combination of features. The *datgen40* data set is a classification problem with 1,000 instances, 40 features, and 10 classes. Note that, this data set was created using Melli's generator of rule-based data sets [22]. Both data sets are available as supplementary material.

For each data set, we incrementally added features and measured the time required to compute all contributions for a single instance. Note that, the number of samples taken $m_{max}$ was such that the probability of having a relative approximation error of more than 1 % was 5 % (relative to the absolute value of the contribution). Adaptive sampling was used, but not quasi-random sampling.

The results in Fig. 2b show that contributions can be computed for these data sets in real-time for a few dozen features, regardless of the choice of the model. The differences between models are in part a consequence of different variances but mostly due to the differences in the time complexity of computing a single prediction, which is the key component of the approximation algorithm's time complexity.

### 4.4 Illustrative examples

We use two types of visualizations: instance visualizations and model visualizations. The former are, as the name suggests, a visualization of the features' local contributions $\varphi$ for a particular instance.

Figure 3 shows a pair of instance visualizations for the same instance from the Monks1 data set but for two different types of models. At the top of an instance visualization are the

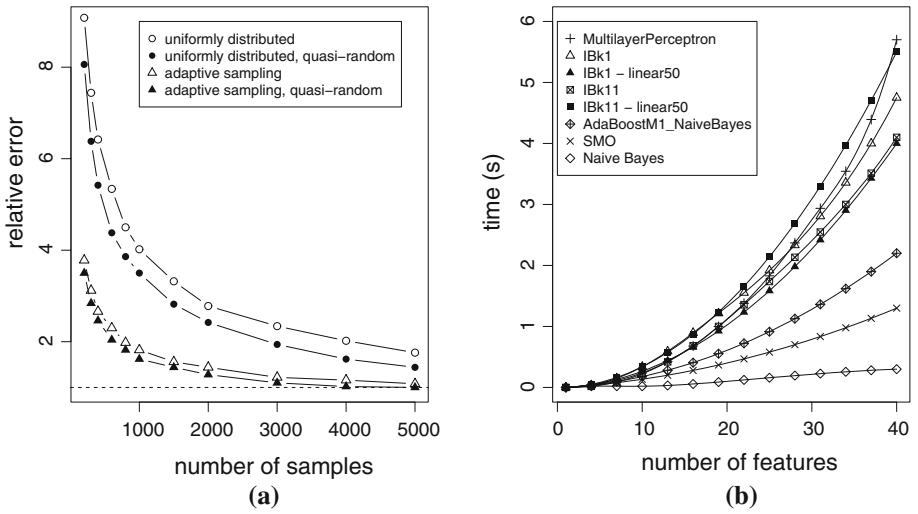


**Fig. 2** The left-hand side plot shows the relative approximation error against the number of samples per feature for four variants of the approximation algorithm. Errors are relative to the error at (5,000× number of input features) samples with both enhancements. The plot on the right hand side shows the running times for computing the contributions for an instance for the datgen40 data set. The most time-consuming models for the linear50 are also included. The remaining model/dataset pairs take less time than the Naive Bayes model and were omitted

name of the data set, the model, the point-of-view class (classification only), the model's prediction for this instance, and the actual (correct) class value. The features' names and values for the instance are listed on the left- and right-hand side, respectively. The boxes contain the features' contributions for this instance. These contributions are also plotted as bars to simplify comparison and identification of features with the largest contribution. Note that, all contributions used in the visualizations were approximated with high precision ($P(\text{error} < 10^{-4}) = 1 - 10^{-3}$).

An instance visualization reveals how individual features contribute to the model's prediction for that instance. For example, for the Monks1 data set, the class value is 1 if (attr1 = attr2 $\vee$ attr5 = 1) and 0 otherwise. The pair of instance visualizations for the same instance from Monks1, but two different models trained on this data set provide us with additional information about how the features influence the models' prediction (see Fig. 3). Although the models are different, the general method facilitates comparison and reveals an important difference between the two models, despite their similar predictions for the same instance. Note that, one-feature-at-a-time approaches would assign a 0 contribution to all features in the artificial neural network case. Perturbing just one feature does not change the model's prediction.

The second pair of instance visualizations is for two different models trained on the cRedundant data set (see Fig. 4). This data set has 5 numerical input features. The class value equals 1 if $A_1 > 0.5$ or $A_2 > 0.7$ or $A_3 < 0.4$. Otherwise, it is 0. Note that, the remaining two features $A_4$ and $A_5$ are copies of $A_1$ and $A_2$, respectively, which introduces some redundancy. For this instance, the values of the first three input features are 0.96, 0.72, and 0.67. The first two features satisfy the condition, while the third does not. Given that both models are successful predictors for this data set, they have learned these concepts, and appropriately, the first two features are assigned a positive contribution, while the third contributes against the class value being 1. Note that, the artificial neural network takes into

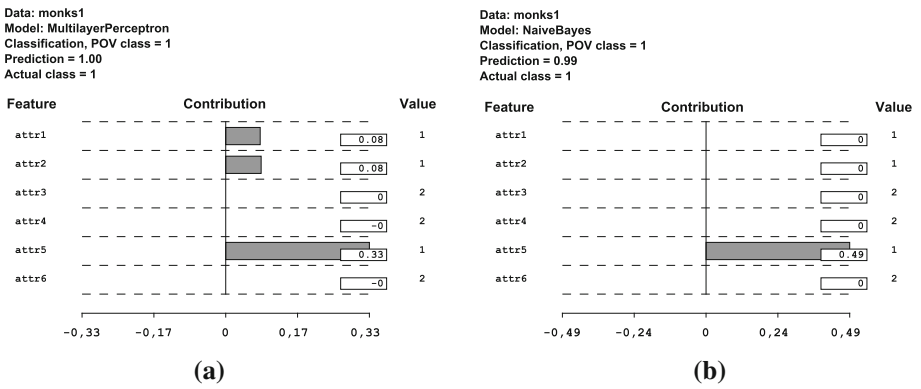


**Fig. 3** The naive Bayes model, due to its assumptions of conditional independence of input features, can not model the importance of the equivalence between attr1 and attr2 (both have a zero contribution). Despite this limitation, it correctly predicts the class value, because for this instance, attr5 = 1 is sufficient (this feature has a substantial positive contribution). The artificial neural network correctly models both concepts

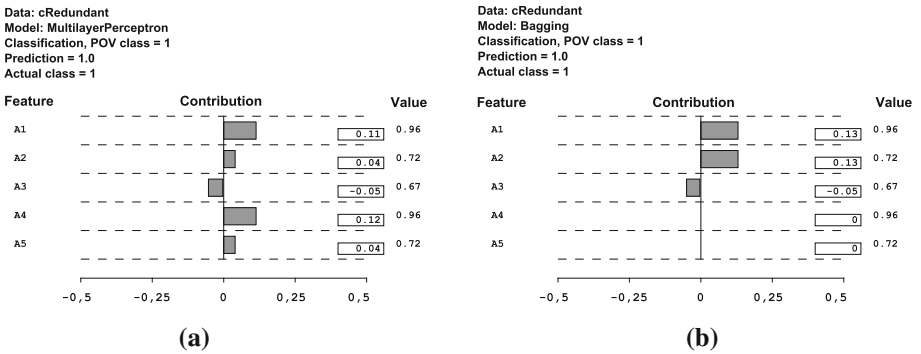


**Fig. 4** Two instance visualizations for the same instance from the cRedundant data set and two different models

account redundant features as well, while bagging with decision trees only takes into account one of each of the input features redundant copies. Although both models are equally good predictors, the explanations are different, because the explanations reveal what the models have learned.

### *4.4.1 Model visualizations*

The second type visualization is the model visualization (see, for example, Fig. 5a). It is composed of *n* marginal effect plots, one for each feature (see Sect. 3.2). For each feature, the mean local contributions are plotted against that feature's value (black line). The importance of each feature (the standard deviation of its contributions) is also included in the form of a gray line.

A model visualization provides us with an overview of how features contribute to the model's predictions. For example, observe Fig. 5a—the model visualization of the decision tree that was trained on the cDisjunctN data set and is good at predicting the class values. The cDisjunct data set is similar to the cRedundant data set; however, the fourth and fifth

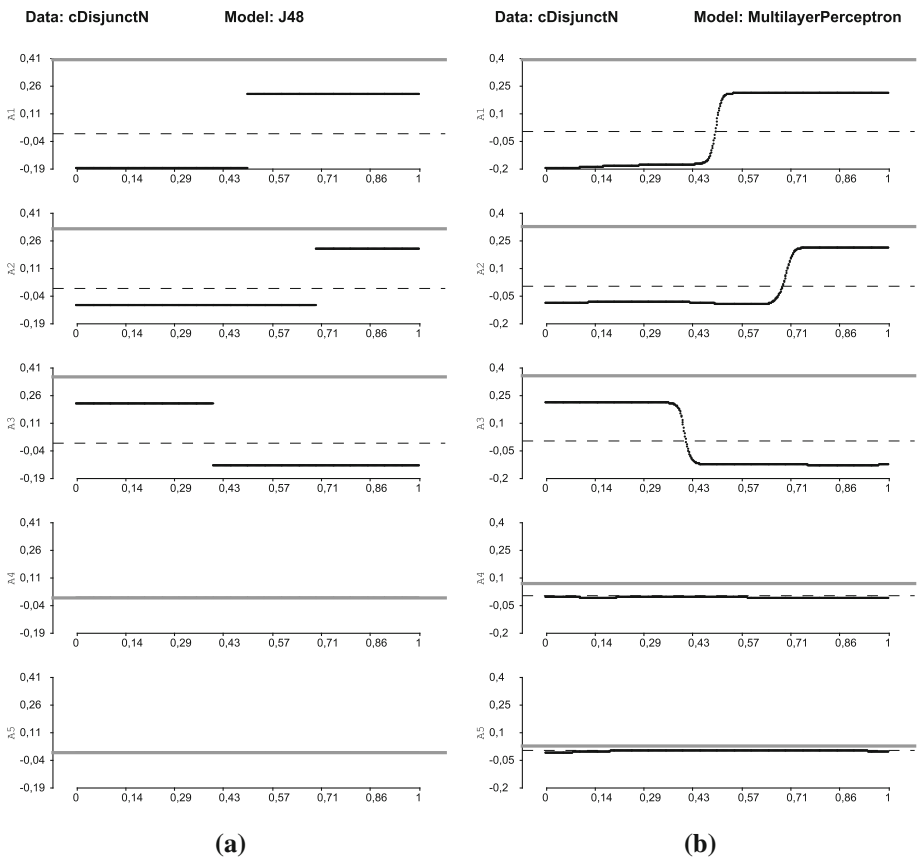


**Fig. 5** Model visualizations for two different models and the cDisjunctN data set. Both models learn the concepts behind the data, and the plotted average contribution functions reveal where the individual features' contribution changes from negative to positive

feature are not copies of the first and second feature. Instead, they are unrelated to the class value. First, the model visualization helps us identify the most important features. The first three features in our example have an equally high importance (gray line—see Sect. 3.2), while the remaining two features are (correctly) identified as of insignificant importance. Second, the plots provide additional information about how features contribute to the model. For example, the first feature (A1) has a negative contribution (speaks against class value 1) if its value is less than 0.5, but contributes positively, if its value is greater than 0.5. Also note that, as shown in Sect. 3.2, if the model is additive, then the plot can also be use to graphically compute the prediction for an arbitrary instance from the data set.

The general method simplifies the comparison of different models or types of models. Figure 5b depicts a model visualization for an artificial neural network trained on the cDisjunct data set. While the performance of both models (see Fig. 5) is similar with respect to prediction quality, the models are slightly different. The visualization reveals the smooth fit of the artificial neural network and characteristically step-function fit of the decision tree. The artificial neural network also slightly overfit the data as the fourth and fifth feature do slightly influence the models predictions.

### 4.5 User-based experiment

The goal of this experiment was to measure if explanations in the form of feature contributions benefit not only (machine learning) experts but also non-expert end-users. Explanations should benefit the user by increasing the user's understanding of the model. The usefulness of explanation methods is usually validated through visual inspection illustrative examples (as we have done in Sect. 4.4) or an application to a real-world problems with domain-expert evaluation.

Only a few studies approach the evaluation in a more general and objective way. [12] compared the usefulness of decision tables, binary decision trees, and decision rules in a study that included 51 post-graduate students. The students had to perform understanding and prediction tasks. The authors measured the prediction accuracy, speed of response, and the level of trust, and concluded that decision tables were most useful. Other studies focused only on decision trees [20] and on decision trees and decision rules [2].

Similar to [12], our goal was to measure the effect of the explanation on the user through the users' performance at prediction tasks, which can be measured objectively. We designed the following experiment:

- the participant is provided, in sequence, with two sets of instances with predictions and unlabeled instances
- for the second set, the participant is also given the situational importance of each feature and all labeled instances,
- for each set separately, the participant is asked to learn what the model does from labeled instances (an explanations, if given) and produces predictions for unlabeled instances.

We constructed two different sets of instances, T1 and T2 (see Table 3), and used two different variants of the experiment. In the first variant (EXP1), the participant is provided with either T1 or T2 (chosen at random). That is, the participant first solves the task without explanations and then the same task with explanations. In the second variant (EXP2), the participant is provided either with T1 first and T2 second or T2 first and T1 second (chosen at random). That is, the participant solves one task without explanations and then a different task with explanations.

A total of 122 computer science students participated in this experiment. The students could only use a pencil and paper to compute their predictions and were limited to 8 minutes per set. All 56 participants in EXP1 were first-year students, which are assumed to have no substantial experience with knowledge discovery. Two groups of students participated in EXP2: 52 1st year students (group A) and 14 4th year students with experience in data mining and knowledge discovery (group B).

For each test instance separately, we ranked the mean-squared errors of the participants predictions. We prefer ranks to actual mean-squared errors to avoid the effect of outliers and facilitate comparison across all four test instances. We tested the statistical significance of the differences with the Wilcoxon test (paired for EXP1 and unpaired for EXP2). We use the 95 % confidence level when determining the significance of the results.

The results are shown in Table 4. Where explanations were provided, prediction errors rank significantly lower across all groups and both variants of the experiment.

Significantly better predictions are a consequence of participants having a better understanding of the model. Given the design of the experiment, it is reasonable to conclude that better understanding was caused by providing explanations. This results suggest that situational importance is a useful form of explanation for non-expert users.

**Table 3** We constructed two sets of learning and testing instances, T1 and T2

| Instance | T1 | | | | T2 | | | |
|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | A3 | C | A1 | A2 | A3 | C |
| learn1 | 11.76 | 70 | 12 | 52 | A | 450 | 21 | 27 |
| $\varphi$ | 0 | 0 | −60 | – | −31 | +11 | 0 | – |
| learn2 | 11.56 | 55 | 16 | 82 | A | 250 | 21 | 2 |
| $\varphi$ | 0 | 0 | −30 | – | −25 | −20 | 0 | – |
| learn3 | 11.86 | 32 | 22 | 127 | B | 280 | 22 | 12 |
| $\varphi$ | 0 | 0 | +15 | – | -7 | −28 | 0 | – |
| learn4 | 12.12 | 80 | 12 | 60 | B | 600 | 20 | 70 |
| $\varphi$ | 0 | 0 | −52 | – | +3 | +20 | 0 | – |
| learn5 | 12.31 | 74 | 28 | 172 | B | 800 | 22 | 75 |
| $\varphi$ | 0 | 0 | +60 | – | +4 | +24 | 0 | – |
| learn6 | 11.65 | 44 | 21 | 120 | C | 250 | 21 | 56 |
| $\varphi$ | 0 | 0 | +8 | – | +31 | −22 | 0 | – |
| learn7 | 11.34 | 72 | 25 | 150 | C | 200 | 19 | 40 |
| $\varphi$ | 0 | 0 | +38 | – | +28 | −35 | 0 | – |
| test1 | 11.91 | 59 | 21 | 120 | A | 600 | 21 | 30 |
| test2 | 11.87 | 28 | 29 | 180 | B | 100 | 23 | 94 |
| test3 | 12.00 | 54 | 27 | 165 | C | 400 | 20 | 2 |
| test4 | 11.73 | 33 | 17 | 90 | D | 800 | 20 | 100 |

Both sets consist of 7 instances with predictions and 4 unlabeled test instances. Explanations are provided in the form of situational importances $\varphi$. Both sets have three input features (A1–A3), numeric class value (C), and have a real-world background. For T1, the input features and class value are cricket length, humidity, temperature, and number of chirps per minute, respectively. The latter is a linear function of temperature, while the remaining two input features are not relevant. For T2, the input features and class value are insecticide type, insecticide amount (in ml), temperature, and percentage of insects killed. The latter depends on insecticide type (C is stronger than B, B is stronger than A) and amount, while temperature is irrelevant. The real-world examples were selected to make the problem less abstract and easier to relate to. Of course, the relationships between the input features and class value were not revealed to the participants

**Table 4** Average ranks for the users' prediction errors for both variants of the experiment and all groups

| | Without | With | $p$ value | N |
|---|---|---|---|---|
| EXP2, -, T1 | 36.00 | 20.00 | $2.2 \times 10^{-16}$ | 28 |
| EXP2, -, T2 | 29.50 | 26.5 | $2.3 \times 10^{-4}$ | 28 |
| EXP2, A, T1 | 30.75 | 21.25 | $6.4 \times 10^{-6}$ | 26 |
| EXP2, A, T2 | 28.25 | 23.75 | $1.5 \times 10^{-2}$ | 26 |
| EXP2, B, T1 | 7.75 | 6.62 | $4.9 \times 10^{-2}$ | 7 |
| EXP2, B, T2 | 8.30 | 5.70 | $1.2 \times 10^{-2}$ | 7 |

$p$ values and group sizes (N) are also provided. Prediction errors without explanations (without) rank higher than when explanations are available (with)

It could be argued that in EXP1, better understanding was not a consequence of providing explanations but of participants performing the same task for the second time when explanations were provided. However, in EXP2, this was controlled for by giving participants a different task when explanations were provided. The only potential threat to the validity of this experiment is the possibility that participants matured . However, given the simplicity of the tasks and the short timespan, we assume that this is unlikely.

## 5 Conclusion

We proposed a general method for explaining how features contribute to classification and regression models' predictions. The method builds on previous work on a general method for computing the situational importance of features for prediction models. By design, the method perturbs all subsets of features to deal with the shortcomings of other existing general methods that do not properly take into account interactions between features.

We derived the mean situational importance of a feature's value, and we show how it can be used as a basis for a model visualization, which provides an overview of how features contribute to the model's predictions. For additive models, this approach generalizes previous additive model-specific methods and general explanation methods.

We also proposed two enhancements to the sampling algorithm (quasi-random and adaptive sampling) that reduce the running time of the algorithm. Empirical results across several types of models and data sets show that the method is an efficient and useful tool for visualizing models, comparing different types of models, and identifying potential errors. Furthermore, an experiment with human participants showed that providing an explanation in the form of feature contributions increased the user's understanding of the prediction model.

## References


1. Achen CH (1982) Intepreting and Using Regression. Sage Publications, Thousand Oaks
2. Allahyari H, Lavesson N (2011) User-oriented assessment of classification model understandability. In: Proceedings of the 11th Scandinavian conference on artificial intelligence, SCAI 2011, pp 11–19
3. Becker B, Kohavi R, Sommerfield D (1997) Visualizing the simple Bayesian classier. KDD workshop on issues in the integration of data mining and data visualization
4. Bhattacharya S, Xu D, Kumar K (2011) An ANN-based auditor decision support system using Benford's law. Decis Support Syst 50(3):576–584
5. Bhattacharyya S, Jha S, Tharakunnel K, Westland JC (2011) Data mining for credit card fraud: a comparative study. Decis Support Syst 50(3):602–613
6. Blanchard J, Guillet F, Briand H (2007) Interactive visual exploration of association rules with rule-focusing methodology. Knowl Inf Syst 13:43–75
7. Castro J, Gómez D, Tejada J (2009) Polynomial calculation of the shapley value based on sampling. Comput Oper Res 36(5):1726–1730
8. De Falco I, Della Cioppa A (2005) An evolutionary approach for automatically extracting intelligible classification rules. Knowl Inf Syst 7:179–201
9. Frank A, Asuncion A (2011) Uci machine learning repository
10. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. SIGKDD Explor Newsl 11(1):10–18
11. Huang Z, Chen H, Hsu CJ, Chen WH, Wu S (2004) Credit rating analysis with support vector machines and neural networks: a market comparative study. Decis Support Syst 37(4):543–558
12. Huysmans J, Dejaeger K, Mues C, Vanthienen J, Baesens B (2011) An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decis Support Syst 51(1):141–154
13. Jaeckel P (2002) Monte Carlo methods in finance. Wiley, New York
14. Jakulin A, Možina M, Demšar J, Bratko I, Zupan B (2005) Nomograms for visualizing support vector machines. KDD '05: 11th ACM SIGKDD, ACM, pp 108–117
15. Kattan MW, Eastham JA, Stapleton AM, Wheeler TM, Scardino PT (1998) A preoperative nomogram for disease recurrence following radical prostatectomy for prostate cancer. J Natl Cancer Inst 90:766–771
16. Knuth DE (1998) The art of computer programming, volume 2: seminumerical algorithms. Addison-Wesley, Boston
17. Kononenko I (1993) Inductive and bayesian learning in medical diagnosis. Appl Artif Intell 7:317–337
18. Lee S (2010) Using data envelopment analysis and decision trees for efficiency analysis and recommendation of B2C controls. Decis Support Syst 49(4):486–497

19. Lemaire V, Feraud R, Voisine N (2008) Contact personalization using a score understanding method. In: International joint conference on neural networks (IJCNN)
20. Lim BY, Dey AK, Avrahami D (2009) Why and why not explanations improve the intelligibility of context-aware intelligent systems. In: Proceedings of the 27th international conference on Human factors in computing systems, CHI '09, ACM, New York, NY, USA, pp 2119–2128
21. Lubsen J, Pool J, van der Does E (1978) A practical device for the application of a diagnostic or prognostic function. Methods Inf Med 17:127–129
22. Melli G (n.d.) The datgen dataset generator. http://www.datasetgenerator.com
23. Možina M, Demšar J, Kattan M, Zupan B (2004) Nomograms for visualization of naive Bayesian classifier. PKDD 2004, Springer, pp 337–348
24. Robnik-Šikonja M, Kononenko I (2008) Explaining classifications for individual instances. IEEE TKDE 20:589–600
25. Shapley LS (1953) A value for n-person games, vol II of Contributions to the theory of games. Princeton University Press, Princeton
26. Szafron D, Poulin B, Eisner R, Lu P, Greiner R, Wishart D, Fyshe A, Pearcy B, Macdonell C, Anvik J (2006) Visual explanation of evidence in additive classifiers. In: Proceedings of innovative applications of artificial intelligence
27. Štrumbelj E, Bosnić Z, Zakotnik B, Grašič-Kuhar C, Kononenko I (2010) Explanation and reliability of breast cancer recurrence predictions. Knowl Inf Syst 24(2):305–324
28. Štrumbelj E, Kononenko I (2010) An efficient explanation of individual classifications using game theory. J Mach Learn Res 11:1–18
29. Štrumbelj E, Kononenko I (2011) A general method for visualizing and explaining black-box regression models. In: Dobnikar A, Lotric U, Ster B (eds) ICANNGA (2), vol 6594 of Lecture notes in computer science. Springer, Berlin, pp 21–30
30. Welford BP (1962) Note on a method for calculating corrected sums of squares and products. Technometrics 4(3):419–420

## Author Biographies

**Erik Štrumbelj** received his Ph.D. in computer science from University of Ljubljana, Slovenia, in 2011. He is currently an assistant professor at the Faculty of Computer and Information Science, University of Ljubljana. His research interests include machine learning and applied statistics.

**Igor Kononenko** received his Ph.D. in 1990 in computer science from University of Ljubljana, Slovenia. He is a professor at the Faculty of Computer and Information Science in Ljubljana. His research interests include artificial intelligence, machine learning, neural networks, and cognitive modeling. He is the (co)author of 200 journal and conference papers and 12 textbooks in these fields.